Theoretical Computer Science 411 (2010) 799–811

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Notions of hyperbolicity in monoids

Michael Hoffmann, Richard M. Thomas *

Department of Computer Science, University of Leicester, Leicester LE1 7RH, UK

ARTICLE INFO

Keywords:
Hyperbolic monoid
Automatic monoid
Biautomatic monoid
Word problem

ABSTRACT

We introduce a notion of hyperbolicity in monoids which is a restriction of that suggested by Duncan and Gilman. One advantage is that the notion gives rise to efficient algorithms for dealing with certain questions; for example, the word problem can be solved in time $\mathcal{O}(n \log n)$. We also introduce a new way of defining automatic monoids which provides a uniform framework for the discussion of these concepts. Hyperbolic monoids (in the sense introduced here) turn out to be biautomatic.

1. Introduction

The notions of hyperbolic [9] and automatic [6] groups have played a fundamental role in computational group theory in recent years. It has been noted (see [15,16] for example) that the definition of automaticity generalizes naturally from groups to semigroups and an exploration of the basic properties of automatic semigroups was undertaken in [4]. There are some issues with this generalization; see [11], for example, where it was shown that the idea generalizes in several non-equivalent ways. Notwithstanding this, a coherent theory of automatic semigroups has been developed, with some fundamental properties (such as the solution of the word problem in quadratic time) generalizing to semigroups.

Whilst the usual definition of automatic lends itself naturally to such a generalization, this has not been the case for hyperbolic. There were several equivalent ways known of defining hyperbolic groups (see [1] for example) but none of these really apply to semigroups. The situation changed with Gilman's elegant characterization of hyperbolic groups in [8] using pushdown automata; this new condition generalizes naturally to the semigroup setting. As a result, Duncan and Gilman [5] proposed this as the definition of a hyperbolic semigroup.

Their definition is entirely natural. One issue, however, is the absence (so far) of efficient algorithms for dealing with hyperbolic semigroups and monoids. It is well known that the word problem for hyperbolic groups can be solved in linear time (even in real time [13]) but the best known algorithm for the word problem in a hyperbolic monoid is exponential [10]. Other questions (such as the conjugacy problem, which can be solved efficiently in hyperbolic groups [7]) are still open as far as hyperbolic monoids are concerned, even as regards decidability.

The purpose of this paper is to show how a restriction of the definition used by Duncan and Gilman in [5] does lead to efficient algorithms. An analysis of Gilman's proof in [8] shows that one can impose restrictions on the pushdown automata used in the definition; we describe these in Section 3. These new definitions are also natural; we point out that it is possible to define automaticity and biautomaticity in terms of pushdown automata (see Remark 13), and these new notions of hyperbolicity arise directly from this observation. An essential part of all this is the definition of a special type of context-free language which we term "sync linear" (see Definition 10). This gives rise to new perspectives on the relationship between hyperbolic and automatic monoids; it enables us to view hyperbolic monoids (at least, in the sense presented here) and automatic monoids in a more uniform fashion than has previously been the case.

A particular aspect of this is the following. It is known that a hyperbolic group is necessarily automatic [6], but this does not generalize to monoids [10]. With the notions of hyperbolicity given here, we recapture this connection; in fact, a monoid

* Corresponding author.
E-mail addresses: mh55@mcs.le.ac.uk (M. Hoffmann), rmt@mcs.le.ac.uk (R.M. Thomas).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.10.016

satisfying one of these new notions is (as is the case in groups) necessarily biautomatic (see Theorem 33); this means that, for example, the conjugacy problem is solvable in such monoids.

In general, the algorithmic properties of this new class of monoids suggest that they are worthy of further study. Whilst the new definitions are equivalent to the previous one in the group setting (see Theorem 24), they allow us to develop efficient algorithms for monoids; for example, the word problem can be solved in time $\mathcal{O}(n \log n)$ (see Theorem 30). Further work is in progress, and it seems that the techniques described here give rise to a number of efficient algorithms for other problems. One interesting feature (which mirrors the developments in automatic monoids) is that, when developing algorithms, the techniques involve formal languages and automata (as opposed to the situation in groups, where the techniques have been more geometric).

We conclude this section by mentioning some notation we will use. For any word α we denote the length of the word α by $|\alpha|$ and the number of occurrences of a symbol x in α by $|\alpha|_x$. For any $k \in \mathbb{N}$, let A^k denote the set of all words α in A^* with $|\alpha| = k$ and $A^{\leqslant k}$ the set of all words α in A^* with $|\alpha| \leqslant k$. Let α^{rev} denote the reversal of the word α. If M is a monoid and $A \subseteq M$ a set of generators of M, then there is a homomorphism $\theta : A^* \to M$ where each α in A^* is mapped to the corresponding element of M. We will be concerned with finite sets A, so that M is finitely generated. In this context, if α and β are elements of A^*, we write $\alpha \equiv \beta$ if α and β are identical as words, and $\alpha = \beta$ if α and β represent the same element of M (i.e. if $\alpha\theta = \beta\theta$).

2. Synchronously regular languages

In this section we describe some aspects of synchronous two-tape finite automata that will be used in our algorithms; we also define some notions of biautomaticity in monoids that we will need later in the paper. The reader is referred to [3,6,14] for background material on formal languages.

If $\alpha \equiv a_1 a_2 \dots a_n$ and $\beta \equiv b_1 b_2 \dots b_m$, we have a finite state automaton with input alphabet $A \times A$ and reading pairs (a_1, b_1), (a_2, b_2), and so on. To deal with the case where $n \neq m$, we introduce a *padding symbol* \$. More formally, we define a mapping

$$\delta^R : A^* \times A^* \to A(2, \$)^*,$$

where $\$ \notin A$ and

$$A(2, \$) = (A \cup \{\$\}) \times (A \cup \{\$\}) - \{(\$, \$)\},$$

by

$$(\alpha, \beta)\delta^R \equiv \begin{cases} (a_1, b_1) \dots (a_n, b_n) & \text{if } n = m \\ (a_1, b_1) \dots (a_n, b_n)(\$, b_{n+1}) \dots (\$, b_m) & \text{if } n < m \\ (a_1, b_1) \dots (a_m, b_m)(a_{m+1}, \$) \dots (a_n, \$) & \text{if } n > m. \end{cases}$$

We have a map that inserts paddings on the left instead of the right; we define

$$\delta^L : A^* \times A^* \to A(2, \$)^*$$

by

$$(\alpha, \beta)\delta^L \equiv ((\alpha^{rev}, \beta^{rev})\delta^R)^{rev}.$$

If α and β have the same length then $(\alpha, \beta)\delta^L$ and $(\alpha, \beta)\delta^R$ coincide; we sometimes just write $(\alpha, \beta)\delta$ in this case.

The following standard facts about regular languages will be useful:

Lemma 1. *If $J \subseteq (A^* \times B^*)\delta^X$ and $K \subseteq (B^* \times C^*)\delta^X$ are regular with $X \in \{L, R\}$, then*

$$\{(\alpha, \gamma)\delta^X : \text{there exists } \beta \in B \text{ with } (\alpha, \beta)\delta^X \in J, (\beta, \gamma)\delta^X \in K\}$$

is regular.

Lemma 2. *Suppose that $K \subseteq (A^* \times A^*)\delta^X$ is regular and $X \in \{L, R\}$. Let α be a word in A^*. If there exists $\beta \in A^*$ such that $(\alpha, \beta)\delta^X \in K$, then such a word β can be found in time $\mathcal{O}(|\alpha|)$.*

Lemma 3. *Let $M = (Q, A(2, \$), \tau, q_0, F)$ be a finite state automaton and $X \in \{L, R\}$. For any word $\alpha \equiv a_1 a_2 \dots a_n \in A^*$ we can create the following collection of sets in time $\mathcal{O}(n)$:*

$$D_i = \left\{ q \in Q : \text{there exists } \beta \in A^* \text{ with } q_0 \xrightarrow{(a_1 \dots a_i, \beta)\delta^X} q \right\}.$$

We also need the following result (see Lemma 4.1 from [11]):

Lemma 4. *Let A be a finite set with $\$ \notin A$ and let $B = A \cup \{\$\}$. Let $\phi : B^* \to A^*$ be the homomorphism defined by*

$$x\phi = \begin{cases} x & \text{for } x \in A \\ \epsilon & \text{for } x \equiv \$. \end{cases}$$

Let L be a regular subset of $(B \times B)^*$. Let k be a constant such that, for all $(a_1, b_1) \dots (a_n, b_n) \in L$, we have

$$|a_1 \dots a_n|_\$ \leqslant k, \qquad |b_1 \dots b_n|_\$ \leqslant k.$$

Then the set

$$K = \{(\alpha, \beta)\delta^R : \alpha \equiv (a_1 \dots a_n)\phi, \beta \equiv (b_1 \dots b_n)\phi \text{ for some } a_i, b_i \in B \text{ with } (a_1, b_1) \dots (a_n, b_n) \in L\}$$

is regular.

Fundamental in the notion of automatic and biautomatic monoids is the concept of "padded" pairs of words. If M is a monoid generated by a finite set A, L is a regular subset of A^* and $a \in A \cup \{\epsilon\}$, then we define:

$${}^{\$}_{a}L = \{(\alpha, \beta)\delta^L : \alpha, \beta \in L, a\alpha = \beta\};$$
$${}^{\$}L_a = \{(\alpha, \beta)\delta^L : \alpha, \beta \in L, \alpha a = \beta\};$$
$${}_aL^{\$} = \{(\alpha, \beta)\delta^R : \alpha, \beta \in L, a\alpha = \beta\};$$
$$L^{\$}_a = \{(\alpha, \beta)\delta^R : \alpha, \beta \in L, \alpha a = \beta\}.$$

Recall that $\alpha a = \beta$ means that αa and β represent the same element of M, not that αa and β are identical as words. Given this, we now recall some notions of biautomaticity in monoids (see [12]):

Definition 5. Let M be a monoid generated by a finite set A and suppose that L is a regular language over A that maps onto M. Let $\bar{A}$ represent $A \cup \{\epsilon\}$. The pair (A, L) is said to be

(i) a *left-biautomatic structure* if ${}^{\$}_{a}L$ and ${}^{\$}L_a$ are regular for $a \in \bar{A}$;
(ii) a *right-biautomatic structure* if ${}_aL^{\$}$ and $L^{\$}_a$ are regular for $a \in \bar{A}$.

A monoid M is said to be *left-biautomatic* if it has a left-biautomatic structure and *right-biautomatic* if it has a right-biautomatic structure.

Remark 6. The point about ${}^{\$}_{\epsilon}L$ (equivalently ${}^{\$}L_\epsilon$) or ${}_\epsilon L^{\$}$ (equivalently $L^{\$}_\epsilon$) being regular is that we can test whether or not two elements of L represent the same element of M. It follows from Lemma 3.7 and Remark 3.8 of [11] that we can assume that L maps bijectively onto M; these languages all become $\{(\alpha, \alpha)\delta : \alpha \in L\}$ in that case, and so are clearly regular.

3. Types of context-free languages

Throughout this paper, if $P = (Q, \Sigma, \Gamma, \tau, q_0, F)$ is a pushdown automaton, we assume there is a special symbol $\perp$ (where $\perp \notin \Gamma$) on the bottom of the stack. This symbol is present at the start of the computation (i.e. the stack only contains $\perp$ initially), is never deleted nor appears anywhere else on the stack. We accept by accept state but our machines all have an "empty stack" (i.e. a stack only containing $\perp$) when a word is accepted. If $(r, \sigma') \in \tau((q, \sigma), \alpha)$ we have a transition from state q to state r reading α where the stack contents change from σ to σ'; we write $(q, \sigma) \xrightarrow{\alpha} (r, \sigma')$. When we refer to the stack contents, we omit $\perp$ unless the stack is empty (in which case we denote the contents by $\perp$); note that the top of the stack appears on the left of the word representing the stack contents.

As the reader will observe, we could have defined acceptance by having both an accept state and an empty stack. However, in our formulation, we have the property that, whenever we are in an accept state, then the stack is empty, and this fact is important in some of our constructions; it turns out that the formulation described here is more convenient for our purposes.

As we mentioned above, we consider restrictions to pushdown automata. To do this, we want to define types of sequences of moves. As above, let $\bar{A}$ denote $A \cup \{\epsilon\}$. Let $O = \{h, y, p\}$ represent the possible operations pus**h**, sta**y**, po**p** to the stack.

Definition 7. Let $(q, \sigma_1) \xrightarrow{\alpha} (r, \sigma_2)$ be a transition in a pushdown automaton P. We say that

$$(a_1, o_1)(a_2, o_2) \dots (a_n, o_n),$$

where $a_i \in \bar{A}$ and $o_i \in O$, is a *trace* of the transition if $\alpha \equiv a_1 \dots a_n$ and there exists a computation path from (q, σ_1) to (r, σ_2) such that the ith step reads a_i and performs the stack operation o_i. (We may have that $a_i \equiv \epsilon$ for some values of i.)

If $\alpha \in L(P)$ we say α has a *trace*

$$t = (a_1, o_1)(a_2, o_2) \dots (a_n, o_n)$$

if there exists $q \in F$ such that there is a transition $(q_0, \perp) \xrightarrow{\alpha} (q, \perp)$ with trace t.

Note that, when defining traces for words, we are only doing so for words accepted by P. We have an analogous concept for a language:

Definition 8. Let $T \subseteq (\bar{A} \times O)^*$. A pushdown automaton is of *type T* if every word accepted has a trace in T; a language L is of *type T* if there is a pushdown automaton of type T accepting L.

A particular example of a class of such languages are the *visibly pushdown languages* described in [2]. In those languages the input set A is partitioned into three pairwise disjoint subsets B, C and D, and the language is of type $(B_h \cup C_y \cup D_p)^*$. However, not every language that is easily described as in Definition 8 is visibly pushdown; for example, it is pointed out in [2] that the language $\{a^n b a^n : n \in \mathbb{N}\}$ is not visibly pushdown.

To reduce the notation needed for types and traces we use the following:

Definition 9. If $B \subseteq \bar{A}$ and $o \in \{h, y, p\}$ we write B_o for $B \times \{o\}$ and b_o for $\{b\} \times \{o\}$. Let

$$(p, \sigma) \xrightarrow[T]{\alpha} (q, \sigma')$$

denote a transition with trace in T which starts in state p with stack contents σ, reads α, and ends in state q with stack contents σ'.

We now introduce a certain kind of context-free language:

Definition 10. A context-free grammar $G = (N, A \cup \{\#\}, R, S)$ (where $\# \notin A$) is said to be *sync linear* if each production rule in R is of the form

$$X \to aYb \quad \text{or} \quad X \to \#$$

where $X, Y \in N$ and $a, b \in A$.

A language K is said to be *sync linear* if it is generated by a sync linear grammar.

Remark 11. If K is a sync linear language (as in Definition 10), then K must be a subset of

$$\{\alpha\#\beta : \alpha, \beta \in A^*, |\alpha| = |\beta|\}. \quad \square$$

The following result will be used throughout the paper:

Lemma 12. *If $\Sigma = A \cup \{\#\}$ and $K \subseteq \Sigma^*$, then the following are equivalent:*

(i) *K is sync linear.*
(ii) *K^{rev} is sync linear.*
(iii) *$K \subseteq \{\alpha\#\beta : |\alpha| = |\beta|\}$ and $\{(\alpha, \beta)\delta : \alpha^{rev}\#\beta \in K\}$ is regular.*
(iv) *$K \subseteq \{\alpha\#\beta : |\alpha| = |\beta|\}$ and $\{(\alpha, \beta)\delta : \alpha\#\beta^{rev} \in K\}$ is regular.*
(v) *K is a context-free language of type $A_h^* \#_y A_p^*$.*

Proof. (i) $\Leftrightarrow$ (ii): This is straightforward; we can convert a sync linear grammar generating K to one generating K^{rev} simply by reversing the right-hand side of each production rule.

(i) $\Rightarrow$ (iv): Let $G = (N, A \cup \{\#\}, S, R)$ be a sync linear grammar generating K. By Remark 11 we have that

$$K \subseteq \{\alpha\#\beta : |\alpha| = |\beta|\}.$$

We construct a finite state automaton $M = (N, A \times A, \tau, S, F)$ accepting $\{(\alpha, \beta)\delta : \alpha\#\beta^{rev} \in K\}$ as follows. We define τ and F by:

$$\tau(X, (a, b)) = Y \quad \text{if } (X \to aYb) \in R;$$
$$F = \{X \in N : (X \to \#) \in R\}.$$

It is now straightforward to check that $L(M) = \{(\alpha, \beta)\delta : \alpha\#\beta^{rev} \in K\}$ since a derivation in G

$$S \Rightarrow a_1 X_1 b_n \Rightarrow a_1 a_2 X_2 b_{n-1} b_n \Rightarrow \cdots \Rightarrow a_1 a_2 \ldots a_n X_n b_1 b_2 \ldots b_n \Rightarrow a_1 a_2 \ldots a_n \# b_1 b_2 \ldots b_n$$

corresponds to a sequence

$$S \xrightarrow{(a_1, b_1)} X_1 \xrightarrow{(a_2, b_2)} \ldots\ldots \xrightarrow{(a_n, b_n)} X_n \in F$$

of transitions accepting $(\alpha, \beta)\delta$ in M.

(iv) $\Rightarrow$ (i): Let $(N, A \times A, \tau, S, F)$ be an finite state automaton accepting the language

$$\{(\alpha, \beta)\delta : \alpha\#\beta^{rev} \in K\}.$$

By assumption, $K \subseteq \{\alpha\#\beta : |\alpha| = |\beta|\}$; so each transition is of the form $\tau(X, (a, b)) = Y$ with $X, Y \in N$ and $a, b \in A$. We let $G = (N, A \cup \{\#\}, S, R)$ be the sync linear grammar with

$$R = \{(X \to aYb) : \tau(X, (a, b)) = Y\} \cup \{(X \to \#) : X \in F\}.$$

It is again straightforward to check that $L(G) = K$.

(iv) $\Leftrightarrow$ (iii): This now follows from (i) $\Leftrightarrow$ (ii) and (i) $\Leftrightarrow$ (iv).

(iii) $\Rightarrow$ (v): Let M be a finite state automaton accepting

$$\{(\alpha, \beta)\delta : \alpha^{rev}\#\beta \in K\};$$

we construct a pushdown automaton P accepting K with P of type $A_{\mathrm{h}}^* \#_{\mathrm{y}} A_{\mathrm{p}}^*$.

P operates in two phases: before reading #, P simply copies the input onto the stack. When P reads # it does not alter the stack. After reading #, P simulates (on each move) the action of reading (a, b) in M where a is the next input in P and b is the symbol on top of the stack. When the input is exhausted (and the stack emptied), P only accepts if we have an accept state in M.

(v) $\Rightarrow$ (iii): Let $P = (Q, A \cup \{\#\}, \Gamma, \tau, s, F)$ be a pushdown automaton of type $A_{\mathrm{h}}^* \#_{\mathrm{y}} A_{\mathrm{p}}^*$ accepting K. Every word in K must be the form $\alpha\#\beta$ with $\alpha, \beta \in A^*$ and, while reading α, P builds up a stack of size $|\alpha|$; then, when reading β, P pops a symbol off the stack for each input read. Since P accepts with empty stack, we must have that $|\alpha| = |\beta|$.

When reading the letters from β (i.e. the letters after #), P essentially operates as a synchronous two-tape finite state automaton where the stack σ serves as the second input tape; so the set of all such possible pairs at each stage forms a regular language. More precisely, if $q \in Q$, then the set

$$C_q = \left\{ (\beta, \sigma)\delta : \text{there exists } p \in F \text{ with } (q, \sigma) \xrightarrow[A_{\mathrm{p}}^*]{\beta} (p, \bot) \right\}$$

is regular. Similarly, for each $q \in Q$, we have that the set

$$D_q = \left\{ (\alpha, \sigma^{rev})\delta : (s, \bot) \xrightarrow[A_{\mathrm{h}}^*\#_y]{\alpha\#} (q, \sigma) \right\}$$

is regular, and so D_q^{rev} is regular. (Note that no word in D_q or D_q^{rev} involves any padding symbols.) Now, by Lemma 1, the set

$$E_q = \{(\alpha^{rev}, \beta)\delta : \text{there exists } \sigma \in \Gamma^* \text{ with } (\alpha^{rev}, \sigma)\delta \in D_q^{rev}, (\beta, \sigma)\delta \in C_q\}$$

is regular. Since there are finitely many states q, the set

$$\{(\alpha^{rev}, \beta)\delta : \text{there exists } q \in Q, \text{ there exists } \sigma \in \Gamma^* \text{ with } (\alpha^{rev}, \sigma)\delta \in D_q^{rev}, (\beta, \sigma)\delta \in C_q\} = \{(\alpha^{rev}, \beta)\delta : \alpha\#\beta \in K\}$$

is regular. □

Remark 13. An advantage of Lemma 12 is that it lets us consider automaticity and biautomaticity in terms of context-free languages. For example, if $L \subseteq A^*$ and we consider

$$L_a^{\$} = \{(\alpha, \beta)\delta^R : \alpha, \beta \in L, \alpha a = \beta\}$$

(as in Definition 5), the regularity of $L_a^{\$}$ is equivalent to

$$K = \{\tilde{\alpha}\#\tilde{\beta}^{rev} : (\alpha, \beta)\delta^R \in L_a^{\$}\}$$

being sync linear, where $\tilde{\alpha}$ and $\tilde{\beta}$ are obtained from α and β by padding the shorter of the two words on the right by symbols \$ to make them of the same length. Another way of saying that $L_a^{\$}$ is regular is to say that K is a context-free language of type $B_{\mathrm{h}}^*\#_{\mathrm{y}}B_{\mathrm{p}}^*$ where $B = A \cup \{\$\}$. □

The next result is reasonably straightforward; we adapt the standard proof via pushdown automata that context-free languages are closed under concatenation and union:

Lemma 14. *If L and K are context-free languages of type T_L and T_K respectively then LK is a context-free language of type $T_L T_K$ and $L \cup K$ is a context-free language of type $T_L \cup T_K$.*

For concatenation, each move out of an accept state f of the first machine M is duplicated as a new move from f to the start state of the second machine N; given that, in our machines, the stack is always empty when accepting a word, this allows us to finish reading a word from L in M and then start reading a word from K in N without introducing any extra moves (so that the new trace is just the concatenation of the two previous traces).

As far as union is concerned, we introduce a new start state t and, for every move out of the start state of M to a state q of M, we introduce a move from t to q with the same label; we proceed similarly for N. The first move in the new machine simulates either the first move of M or the first move of N and, thereafter, we proceed entirely within M or entirely within N (so that the new set of traces is just the union of the two previous sets of traces).

It is also known that one can insert a context-free language into another to yield a context-free language; modifying the proof slightly gives the following result:

Lemma 15. *Suppose that $L \subseteq A^*\{\#\}A^*$ (where $\# \notin A$) is a context-free language of type $W_1\#_y W_2$ with $W_1, W_2 \subseteq (\bar{A} \times \{h, y, p\})^*$ and that K is a context-free language of type T_K. Then*

$$L' = \{\alpha\beta\gamma : \alpha\#\gamma \in L, \beta \in K\}$$

is a context-free language of type $W_1 T_K W_2$.

The proof again is fairly straightforward; instead of reading # the machine performs a computation reading a word from K. We may assume (without loss of generality) that the stack alphabets of the machines accepting K and L are disjoint; in this way, whatever symbol happens to be on the top of the stack when we reach # then serves as the bottom symbol in the computation reading the word from K. Again, the fact that, in our machines, the stack is always empty when accepting a word is important; having read the word from K, the stack has been restored to that which existed in the original machine when one reached #, and the original computation then proceeds as before.

Another result in a similar vein is the following:

Lemma 16. *If $L \subseteq A^*$ is a context-free language of type T and $K \subseteq A^*$ is a regular language then $L \cap K$ is a context-free language of type T.*

The proof of the fact that the intersection of a regular language and a context-free language is context-free (tagging the states of the pushdown automaton with the states of a finite automaton) goes through unchanged here.

The following result, which allows us to change the type of a language, is a little technical but will be useful in what follows:

Lemma 17. *If $B \subseteq A$, $W_1, W_2 \in (\{A \cup \{\epsilon\}) \times \{h, y, p\} - B_y)^*$ and $L \subseteq A^*$ is a context-free language of type $W_1 B_y W_2$, then L is also of type $W_1 B_h W_2 \epsilon_p$.*

Proof. Let P be a pushdown automaton accepting L of type $W_1 B_y W_2$; we construct a new pushdown automaton P' of the desired type. Loosely speaking, after the additional push in P', we need to look at the top two elements of the stack. We do this in the obvious way by changing the stack alphabet.

Let $Q' = (Q \times \{1, 2\}) \cup \{f\}$; f will be the new accept state which we will reach after reading ϵ and popping the last element off the stack at the end of the input. We encode two elements of the old stack symbols into one; so we change the stack alphabet to $\Delta = (\Gamma \cup \{\perp\}) \times (\Gamma \cup \{\perp\})$.

To begin with P' behaves similarly to P:

$((q, 1), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 1), \sigma)$ if $(q, d_1\beta) \xrightarrow{a} (r, \beta)$ in P;
$((q, 1), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 1), (d_1, d_2)\sigma)$ if $(q, d_1\beta) \xrightarrow{a} (r, d_1\beta)$ in P and $a \notin B$;
$((q, 1), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 1), (d, d_1)(d_1, d_2)\sigma)$ if $(q, d_1\beta) \xrightarrow{a} (r, dd_1\beta)$ in P.

Here σ represents an element of Δ^* and β an element of Γ^*. In addition, for an empty stack, we have similar transitions:

$((q, 1), \perp) \xrightarrow{a} ((r, 1), \perp)$ if $(q, \perp) \xrightarrow{a} (r, \epsilon)$ in P and $a \notin B$;
$((q, 1), \perp) \xrightarrow{a} ((r, 1), (d', \perp))$ if $(q, \perp) \xrightarrow{a} (r, d')$ in P.

We change to a new mode after P has read an element of B whilst not altering the stack. We first have the following transitions:

$((q, 1), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 2), (d_1, d_1)(d_1, d_2)\sigma)$ if $(q, d_1\beta) \xrightarrow{a} (r, d_1\beta)$ in P and $a \in B$;
$((q, 1), \perp) \xrightarrow{a} ((r, 2), (\perp, \perp))$ if $(q, \perp) \xrightarrow{a} (r, \perp)$ in P and $a \in B$.

From this point on P' essentially operates on the second component of the top stack symbol:

$((q, 2), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 2), \sigma)$ if $(q, d_2\beta) \xrightarrow{a} (r, \beta)$ in P;
$((q, 2), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 2), (d_1, d_2)\sigma)$ if $(q, d_2\beta) \xrightarrow{a} (r, d_2\beta)$ in P;
$((q, 2), (d_1, d_2)\sigma) \xrightarrow{a} ((r, 2), (d_1, d')(d_1, d_2)\sigma)$ if $(q, d_2\beta) \xrightarrow{a} (r, d'd_2\beta)$ in P.

Finally, to simulate the empty stack of P in P', we have:

$((q, 2), (d_1, \perp)) \xrightarrow{a} ((r, 2), (d_1, \perp)$ if $(q, \perp) \xrightarrow{a} (r, \perp)$ in P;
$((q, 2), (d_1, \perp)) \xrightarrow{a} ((r, 2), (d_1, d')(d_1, \perp))$ if $(q, \perp) \xrightarrow{a} (r, d')$ in P.

The computation in P' has been essentially the same as in P except for the fact that we performed a push operation in the middle whilst reading an element of B (as opposed to leaving the stack unaltered at that point). To accept with our stack empty, we now need to clear a single element off the stack at the end of the computation (without reading any input) and move to our accept state f. We have the following transitions:

$((q, 2), (d_1, \perp)) \xrightarrow{\epsilon} (f, \perp)$ if q is an accept state of P.

Since there are no transitions from the new accept state f, this ϵ_p move can only be done at the end of an accepting computation in P'. □

In a similar vein we have the following result:

Lemma 18. *If $B \subseteq A$, $W_1, W_2, W_3 \in (\{A \cup \{\epsilon\}) \times \{h, y, p\} - B_y)^*$ and $L \subseteq A^*$ is a context-free language of type $W_1 B_y W_2 \#_y W_3$ (where $\# \notin B$), then L is also of type $W_1 B_h W_2 \#_y \epsilon_p W_3$.*

4. Hyperbolic structures

In this section we introduce our notions of hyperbolicity; as we explained in Section 1, these are obtained by following the definition given in [5] but imposing constraints on the type of the pushdown automaton. The three types we will consider are:

$$T_1 = A_h^* \#_y A_p^* A_y^{\leqslant 1} A_h^* \#_y A_p^*;$$
$$T_2 = A_h^* \#_y A_p^* A_h^* \#_y A_p^* (\epsilon_p^* \cup A_y^*);$$
$$T_3 = A_h^* \#_y A_p^* A_h^* \#_y (\epsilon_p^* \cup A_y^*) A_p^*.$$

In [5], if A is a finite generating set for a monoid M, then Duncan and Gilman refer to a regular language L over A mapping onto M as being a "hyperbolic structure" for M if the language

$$\{\alpha\#\beta\#\gamma^{rev} : \alpha, \beta, \gamma \in L, \alpha\beta = \gamma\}$$

is context-free. Given this, we now make the following definition:

Definition 19. A monoid M is called *T_i-hyperbolic* if M has a hyperbolic structure (A, L) such that

$$\{\alpha\#\beta\#\gamma^{rev} : \alpha, \beta, \gamma \in L, \alpha\beta = \gamma\}$$

is of type T_i; (A, L) is then a *T_i-hyperbolic structure* for M.

We will refer to the language $\{\alpha\#\beta\#\gamma^{rev} : \alpha, \beta, \gamma \in L, \alpha\beta = \gamma\}$ as L_{hyp} for the remainder of this paper.

Not all monoids which are hyperbolic in these new ways are close to being groups; consider the following example:

Example 20. Let M be the monoid defined by the presentation

$$\langle a, b, x : xa^i x = xb^i x \text{ for } i > 0\rangle.$$

M is neither finitely presented nor cancellative; however we can show that M is T_1-hyperbolic.

Let $A = \{a, b, x\}$ and $L = A^* - A^*\{x\}\{b\}^*\{x\}A^*$; then, for all $\alpha, \beta \in L$, either $\alpha\beta \equiv \gamma \in L$ or $\alpha \equiv \alpha_1 x b^i$ and $\beta \equiv b^j x \beta_2$ for some $i + j > 0$ and $\alpha_1, \beta_2 \in A^*$. In the latter case

$$\alpha\beta = \alpha_1 x a^{i+j} x \beta_2 \equiv \gamma \in L.$$

A pushdown automaton that pushes all the elements of α and β onto the stack can verify that γ is of the required form while reading γ^{rev} and popping a symbol off the stack for each symbol of γ^{rev}. □

The following result follows directly from Lemma 17:

Lemma 21. *If M is a T_1-hyperbolic monoid then M is also T_2-hyperbolic.*

In a similar fashion, given Lemma 18, we have:

Lemma 22. *If M is a T_1-hyperbolic monoid, then M is also T_3-hyperbolic.*

In fact, using the techniques developed in this paper, one can show:

Lemma 23. *A monoid M is T_3-hyperbolic if and only if M^{rev} is T_2-hyperbolic. Furthermore, (A, L) is a T_3-hyperbolic structure for M if and only if (A, L^{rev}) is a T_2-hyperbolic structure for M^{rev}.*

We explain how Lemma 23 follows from the techniques developed here in Remark 26 below.

Given Lemma 23, we will focus on T_2-hyperbolic monoids. As we explained in the introduction, part of the motivation for these notions springs from the following:

Theorem 24. *Let M be a group and let $1 \leqslant i \leqslant 3$; then M is hyperbolic if and only if M is T_i-hyperbolic.*

Proof. "⇒": Let M be a hyperbolic group generated by a set A; given Lemmas 21 and 23, it is sufficient to show that M is T_1-hyperbolic.

Each element in M is represented by several words in A^*; we are only interested in, for any given element, the representatives of minimum length. Let L be the set of all such words (so that, if $\alpha \in L$, $\beta \in A^*$ and $\alpha = \beta$, then $|\alpha| \leqslant |\beta|$; such words α label geodesics in the Cayley graph of M). It is well known that, for a hyperbolic group, this set L is regular. In addition, Gilman's characterization of hyperbolic groups in [8] shows that the language L_{hyp} is context-free. His proof proceeds via a context-free grammar G which can be taken to be of the following form. The set of non-terminals N is the disjoint union of sets X (which contains the sentence symbol), Y and Z; the production rules are of the form:

$$X_i \to aX_jb; \qquad X_i \to Y_kZ_l; \qquad X_i \to Y_kcZ_l;$$
$$Y_i \to aY_jb; \qquad Y_i \to \#; \qquad Z_i \to aZ_jb; \qquad Z_i \to \#,$$

where $a, b, c \in A, X_\ell \in X, Y_\ell \in Y$ and $Z_\ell \in Z$. We will build $L = L(G)$ out of smaller components of particular types; we then assemble these components and show that L_{hyp} is of type T_1.

Let $L_{Y_i} = \{\eta \in A^* : Y_i \overset{*}{\Rightarrow} \eta\}$ and $L_{Z_i} = \{\eta \in A^* : Z_i \overset{*}{\Rightarrow} \eta\}$; these are sync linear and, by Lemma 12, are of type $A_{\mathrm{h}}^*\#_{\mathrm{y}}A_{\mathrm{p}}^*$. By Lemma 14, for any Y_i and Z_j and any $c \in A$, the languages $L_{Y_i}L_{Z_j}$ and $L_{Y_i}cL_{Z_j}$ are of type $A_{\mathrm{h}}^*\#_{\mathrm{y}}A_{\mathrm{p}}^*A_{\mathrm{y}}^{\leqslant 1}A_{\mathrm{h}}^*\#_{\mathrm{y}}A_{\mathrm{p}}^*$.

Let G' be the context-free grammar with non-terminals X, the same starting symbol as G and the following transitions:

$$X_k \to aX_lb \quad \text{if } X_k \to aX_lb \text{ is a transition in } G;$$
$$X_k \to \#_{i,c,j} \quad \text{if } X_k \to Y_icZ_j \text{ is a transition in } G;$$
$$X_k \to \#_{i,j} \quad \text{if } X_k \to Y_iZ_j \text{ is a transition in } G.$$

Let

$$L_{i,c,j} = L(G') \cap A^*\{\#_{i,c,j}\}A^* \text{ and } L_{i,j} = L(G') \cap A^*\{\#_{i,j}\}A^*.$$

By Lemmas 12 and 16, each of $L_{i,j}$ and $L_{i,c,j}$ is of type $A_{\mathrm{h}}^*B_{\mathrm{y}}A_{\mathrm{p}}^*$ with B the set of all the symbols $\#_{i,j}$ and $\#_{i,c,j}$. If we replace $\#_{i,c,j}$ in $L_{i,c,j}$ with $L_{Z_i}\{c\}L_{Y_j}$ we get $K_{i,c,j}$, and $K_{i,j}$ is obtained in a similar fashion:

$$K_{i,c,j} = \{\eta\xi\zeta : \eta\#_{i,c,j}\zeta \in L_{i,c,j},\ \xi \in L_{Z_i}\{c\}L_{Y_j}\},$$
$$K_{i,j} = \{\eta\xi\zeta : \eta\#_{i,j}\zeta \in L_{Z_i}L_{Y_j},\ \xi \in L_{i,j}\}.$$

By Lemma 15, $K_{i,c,j}$ and $K_{i,j}$ are of type T_1. Since, by construction,

$$L = L(G) = \bigcup K_{i,c,j} \cup \bigcup K_{i,j},$$

we have, by Lemma 14, that L is of type T_1 as required.

"$\Leftarrow$": If M is a group with T_i-hyperbolic structure (A, L) then the set L_{hyp} is a context-free language; so M is hyperbolic by [8]. □

5. Word problem of T_i-hyperbolic monoids

Given a monoid with a T_2-hyperbolic structure (A, L) we will show that the word problem is solvable in time $\mathcal{O}(n \log(n))$. Our first aim is perform a "multiplication" of two words in L into a word in L in linear time.

Let $P = (Q, A \cup \{\#\}, \Gamma, \tau, q_o, F)$ be a pushdown automaton of type T_2 with $L(P) = L_{hyp}$. We are particularly interested at what happens when we read the # symbols; we think of a triangle with sides labelled by α, β and γ^{rev}, and talk about the "corners" of the triangle. Let Γ' denote $\Gamma \cup \{\bot\}$. For $\mu, \nu \in A^*$ and $\sigma \in \Gamma^*$ let

$$T_{\mu,\nu,\sigma} = \left\{(p, q, t) \in Q \times Q \times \Gamma' : (p, t\sigma) \xrightarrow[A_{\mathrm{h}}^*\#_{\mathrm{y}}A_{\mathrm{p}}^*]{\mu\#\nu} (q, t\sigma)\right\}.$$

If $t \equiv \bot$ we must have $\sigma \equiv \epsilon$ (this convention applies to similar situations in the remainder of the paper). Here μ represents a suffix of α and ν a prefix of β.

The trace of the transition specifies that elements will be pushed on the stack, followed by a stay operation, and then elements will be popped off the stack. Since the end configuration has the same stack as the initial one, P has performed the same number of pushes as pops. The element t will never be removed from the stack during the transition and therefore the set is independent of σ; so, from now on, we will omit σ and denote this set by $T_{\mu,\nu}$. Each such set $T_{\mu,\nu}$ is a subset of $Q \times Q \times \Gamma'$ and is therefore bounded in size by the choice of P.

When dealing with these sets we want to be able to construct $T_{a\mu,\nu b}$ out of $T_{\mu,\nu}$; this can be done in the following way:

$$T_{a\mu,\nu b} = \Big\{(p, q, t) \in Q \times Q \times \Gamma' : \text{there exists } (p', q', t') \in T_{\mu,\nu} \text{ with } (p, t\sigma) \xrightarrow[A_{\mathrm{h}}]{a} (p', t't\sigma) \text{ and } (q', t't\sigma) \xrightarrow[A_{\mathrm{p}}]{b} (q, t\sigma)\Big\}.$$

If $t \equiv \bot$, then we have

$$(p, t\sigma) = (p, \bot) \xrightarrow[A_h]{a} (p', t') \quad \text{and} \quad (q', t') \xrightarrow[A_p]{b} (q, \bot);$$

again, we adopt a similar convention for the remainder of the paper. This enables us to create a complete deterministic finite state automaton M_T where each state corresponds to a subset of $Q \times Q \times \Gamma'$; the input alphabet is $A \times A$ and $\tau_T(s_T, (\mu^{rev}, \nu)\delta) = T_{\mu,\nu}$.

For any given $p, q \in Q$ and $t \in \Gamma'$ we can choose the accept states of M to be all states which contain (p, q, t). Hence the set

$$C_{p,q,t} = \left\{ (\mu^{rev}, \nu)\delta : (p, t\sigma) \xrightarrow[A_h^* \# _y A_p^*]{\mu \# \nu} (q, t\sigma) \right\} = \{(\mu^{rev}, \nu)\delta : (p, q, t) \in T_{\mu,\nu}\}$$

is regular. In terms of our triangle, the sets $C_{p,q,t}$ are relevant when considering the corner between α and β. We will now give similar arguments to define a deterministic complete finite state automaton and regular set for each of the other two corners.

First consider the corner between β and γ. For $\mu, \nu \in A^*$ and $\sigma \in \Gamma^*$ let

$$V_{\mu,\nu,\sigma} = \left\{ (p, q, t) \in Q \times Q \times \Gamma' : (p, t\sigma) \xrightarrow[A_h^* \# _y A_p^* \epsilon_p^*]{\mu \# \nu} (q, t\sigma) \right\}.$$

Again these sets are independent of σ and we can build the sets up. Here μ represents a suffix of β and ν a prefix of γ^{rev}. Since μ could be longer than ν, we have to distinguish between the following two cases:

for $|\mu| = |\nu|$: $V_{b\mu,\nu c} = \{(p, q, t) \in Q \times Q \times \Gamma' :$ there exists $(p', q', t') \in V_{\mu,\nu}$ with $(p, t\sigma) \xrightarrow[A_h]{b} (p', t't\sigma)$, $(q', t't\sigma) \xrightarrow[A_p]{c} (q, t\sigma)\}$;

for any μ, ν ($|\mu| \geqslant |\nu|$): $V_{b\mu,\nu} = \{(p, q, t) \in Q \times Q \times \Gamma' :$ there exists $(p', q', t') \in V_{\mu,\nu}$ with $(p, t\sigma) \xrightarrow[A_h]{b} (p', t't\sigma)$, $(q', t't\sigma) \xrightarrow[\epsilon_p]{\epsilon} (q, t\sigma)\}$.

This leads to a complete deterministic finite state automaton M_V with two sorts of transition depending whether or not a padding symbol has already been used. The states of M_V are subsets of $Q \times Q \times \Gamma'$, the alphabet is $A(2, \$)$ and $\tau_V(s_V, (\mu^{rev}, \nu)\delta^R) = V_{\mu,\nu}$. The padding symbol \$ used in M_V corresponds to an ϵ_p move in P, and it is encoded in Q as to whether or not this has taken place.

As before, for any $p, q \in Q$ and $t \in \Gamma'$, we can set the accept states in M_V to be all states that contain (p, q, t); so the following set is regular:

$$E_{p,q,t} = \{(\mu^{rev}, \nu)\delta^R : (p, q, t) \in V_{\mu,\nu}\}.$$

We now use similar arguments for the corner between α and γ. For $\mu, \nu \in A^*$ let

$$U_{\mu,\nu} = \left\{ (p, q, t) \in Q \times Q \times \Gamma' : (q_0, \bot) \xrightarrow[A_h^*]{\mu} (p, t\sigma), \; (q, t\sigma) \xrightarrow[A_p^*(A_y^* \cup \epsilon_p^*)]{\nu^{rev}} (q_f, \bot) \text{ for some } q_f \in F \text{ for some } \sigma \in \Gamma^* \right\}.$$

Again we can build the sets up. Here μ represents a prefix of α and ν a prefix of γ. However due to the fact that we can either clear the stack with empty moves or else read the rest of γ^{rev} whilst the stack is empty, we have to distinguish three cases. Let $\mu, \nu \in A^*$ and $a, c \in A$; then:

$$U_{\mu a,\nu c} = \left\{ (p, q, t) : \text{there exists } (p', q', t') \in U_{\mu,\nu} \text{ with } (p', t'\sigma) \xrightarrow[A_h]{a} (p, tt'\sigma), (q', tt'\sigma) \xrightarrow[A_p]{c} (q, t'\sigma) \right\};$$

$$U_{\mu a,\epsilon} = \left\{ (p, q, t) : \text{there exists } (p', q', t') \in U_{\mu,\epsilon} \text{ with } (p', t'\sigma) \xrightarrow[A_h]{a} (p, tt'\sigma), (q', tt'\sigma) \xrightarrow[\epsilon_p]{\epsilon} (q, t'\sigma) \right\};$$

$$U_{\epsilon,\nu c} = \left\{ (q_0, q, \bot) : \text{there exists } (q_0, q', \bot) \in U_{\epsilon,\nu} \text{ with } (q', \bot) \xrightarrow[A_y]{c} (q, \bot) \right\}.$$

This leads to a deterministic complete finite state automaton M_U over the alphabet $A(2, \$)$ with states $Q \times Q \times \Gamma'$ and transitions $\tau_U(s_U, (\mu, \nu)\delta^L) = U_{\mu,\nu}$. As before, for any $p, q \in Q$ and $t \in \Gamma'$, we can set the accept states to be all states that contain (p, q, t); therefore the following set is regular:

$$D_{p,q,t} = \{(\mu, \nu)\delta^L : (p, q, t) \in U_{\mu,\nu}\}.$$

The main step in solving the word problem is now the following result:

Lemma 25. *Let M be a monoid with a T_2-hyperbolic structure (A, L). Given $\alpha, \beta \in L$, a word $\gamma \in L$ with $\alpha\beta = \gamma$ can be constructed in time $\mathcal{O}(|\alpha| + |\beta|)$.*

Proof. Let $P = (Q, A \cup \{\#\}, \Gamma, \tau, q_0, F)$ be a pushdown automaton of type T_2 accepting L_{hyp}. Assume that $\alpha \equiv a_1 a_2 \dots a_n$ and $\beta \equiv b_1 b_2 \dots b_m$ are given; we want to construct $\gamma \in L$ with $\gamma = \alpha\beta$. The algorithm will work in two steps; the figure below indicates some of the notation used.

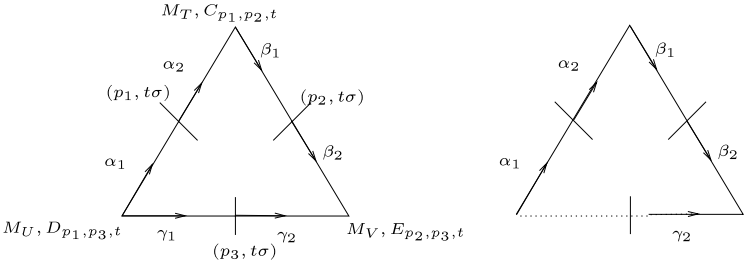

Fig. 1. Notation used.

Goal 1: Find i, p_1, p_2, p_3 and t such that there exist $\gamma_1, \gamma_2 \in A^*, \sigma \in \Gamma^*$ and $q_f \in F$ with either:

$$(q_0, \bot) \xrightarrow[A_h^*]{\alpha_1} (p_1, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\alpha_2\#\beta_1} (p_2, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\beta_2\#\gamma_2^{rev}} (p_3, t\sigma) \xrightarrow[A_p^*(\epsilon_p^* \cup A_y^*)]{\gamma_1^{rev}} (q_f, \bot)$$

$$\text{or: } (q_0, \bot) \xrightarrow[A_h^*]{\alpha_1} (p_1, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\alpha_2\#\beta_1} (p_2, t\sigma) \xrightarrow[A_h^*\#_y A_p^*\epsilon_p^+]{\beta_2\#\gamma_2^{rev}} (p_3, t\sigma) \xrightarrow[\epsilon_p^*]{\epsilon} (q_f, \bot)$$

where $\alpha \equiv \alpha_1\alpha_2$ with $|\alpha_2| = i$ and $\beta \equiv \beta_1\beta_2$ with $|\beta_1| = i$.

Goal 2: Create $\gamma \in L$ with $\gamma = \alpha\beta$.

We now describe the steps of our algorithm that allow us to achieve these goals.

Step 1: Let G_j be the set of states which M_U can be in for any input of the form $(a_1 a_2 \dots a_j, \gamma_1)\delta^L$ with $\gamma_1 \in A^*$ and I_k the set of states that M_V can be in for any input $(b_m \dots b_{m-k+1}, \gamma_2)\delta^R$ with $\gamma_2 \in A^*$. Let H_i be $T_{a_n \dots a_{n-i+1}, b_1 \dots b_i}$.

By Lemma 3 all of these sets can be created in time $\mathcal{O}(|\alpha| + |\beta|)$. The algorithm will now find the least i such that there exists $p_1, p_2, p_3 \in Q, t \in \Gamma, I \in I_{m-i}, G \in G_{n-i}$ and $H \in H_i$ such that $(p_1, p_2, t) \in H$, $(p_2, p_3, t) \in I$ and $(p_1, p_3, t) \in G$. Since the sizes of all sets H_i, G_j and I_k can be uniformly bounded in terms of P, the check for any particular i is done in constant time; hence we obtain i in time $\mathcal{O}(|\alpha| + |\beta|)$.

Step 2: From Step 1 we have determined i, p_1, p_2, p_3 and t. Let $\alpha_1\alpha_2 \equiv \alpha$ with $|\alpha_2| = i$ and $\beta_1\beta_2 \equiv \beta$ with $|\beta_1| = i$.

Since $D_{p_1,p_3,t}$ is regular we can find γ_1 such that $(\alpha_1, \gamma_1)\delta \in D_{p_1,p_3,t}$ in time $\mathcal{O}(n - i)$ by Lemma 2. Similarly, $E_{p_2,p_3,t}$ is regular and we can find γ_2 such that $(\beta_2^{rev}, \gamma_2^{rev})\delta^L \in E_{p_2,p_3,t}$ in time $\mathcal{O}(m - i)$ by Lemma 2. So we have that either:

$$(q_0, \bot) \xrightarrow[A_h^*]{\alpha_1} (p_1, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\alpha_2\#\beta_1} (p_2, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\beta_2\#\gamma_2^{rev}} (p_3, t\sigma) \xrightarrow[A_p^*(\epsilon_p^* \cup A_y^*)]{\gamma_1^{rev}} (q_f, \bot)$$

$$\text{or: } (q_0, \bot) \xrightarrow[A_h^*]{\alpha_1} (p_1, t\sigma) \xrightarrow[A_h^*\#_y A_p^*]{\alpha_2\#\beta_1} (p_2, t\sigma) \xrightarrow[A_h^*\#_y A_p^*\epsilon_p^+]{\beta_2\#\gamma_2^{rev}} (p_3, t\sigma) \xrightarrow[\epsilon_p^*]{\epsilon} (q_f, \bot)$$

for some $q_f \in F$ and $\sigma \in \Gamma^*$. Hence we have that

$$\alpha\#\beta\#(\gamma_1\gamma_2)^{rev} \in L(P)$$

and $\gamma \equiv \gamma_1\gamma_2 = \alpha\beta$ with so $\gamma \in L$ as required. □

Remark 26. We are now in a position to establish Lemma 23.

We have seen how a T_2-hyperbolic structure gives rise to sets of regular languages

$$\{C_{p,q,t} : p, q \in Q, t \in \Gamma'\},$$
$$\{D_{p,q,t} : p, q \in Q, t \in \Gamma'\},$$
$$\{E_{p,q,t} : p, q \in Q, t \in \Gamma'\}$$

corresponding to the corners of the triangle in Fig. 1. However, given such sets of regular languages, we can use them to reconstruct our T_2-hyperbolic structure

$$L_{hyp} = \bigcup_{p_1,p_2,p_3 \in Q, t \in \Gamma'} \{\alpha_1\alpha_2\#\beta_1\beta_2\#\gamma_2^{rev}\gamma_1^{rev} : (\alpha_1, \gamma_1)\delta^L \in D_{p_1,p_3,t}, (\alpha_2^{rev}, \beta_1)\delta \in C_{p_1,p_2,t}, (\beta_2^{rev}, \gamma_2^{rev})\delta^R \in E_{p_2,p_3,t}\}.$$

We now consider the following sets:

$$K_{p_1,p_2,p_3,t} = \{\beta_2^{rev}\beta_1^{rev}\#\alpha_2^{rev}\alpha_1^{rev}\#\gamma_1\gamma_2 : (\alpha_1,\gamma_1)\delta^L \in D_{p_1,p_3,t}, (\alpha_2^{rev},\beta_1)\delta \in C_{p_1,p_2,t}, (\beta_2,\gamma_2)\delta^R \in E_{p_2,p_3,t}\}.$$

For any fixed $p_1, p_2, p_3 \in Q$ and $t \in \Gamma'$ we now show that the set $K_{p_1,p_2,p_3,t}$ is context-free and establish its type.

There are two cases to consider.

Case 1: We first consider the case where

$$E_{p_1,p_2,p_3,t} \subseteq (A \times A)^*.$$

In this case no padding symbols are used and so $(\beta_2,\gamma_2)\delta^R = (\beta_2,\gamma_2)\delta$ for all $(\beta_2,\gamma_2)\delta^R \in E_{p_2,p_3,t}$.

Since the set $C_{p_1,p_2,p_3,t}$ is regular the set

$$\{\beta_1^{rev}\#\alpha_2^{rev} : (\alpha_2^{rev},\beta_1)\delta \in C_{p_1,p_2,t}\}$$

is of type $A_h^*\#_y A_p^*$ by Lemma 12. Similarly the set

$$\{\alpha_1^{rev}\#\gamma_1 : (\alpha_1,\gamma_1)\delta^L \in D_{p_1,p_3,t}\}$$

is of type $A_h^*\#_y(\epsilon_p^* \cup A_y^*)A_p^*$ and the set

$$\{\beta_2^{rev}\#\gamma_2 : (\beta_2,\gamma_2)\delta \in E_{p_2,p_3,t}\}$$

is of type $A_h^*\#_y A_p^*$.

By Lemmas 14 and 15 we have that the set $K_{p_1,p_2,p_3,t}$ is of type $A_h^*A_h^*\#_y A_p^*A_h^*\#_y(\epsilon_p^* \cup A_y^*)A_p^*$.

Case 2: Let us now assume that $E_{p_1,p_2,p_3,t}$ contains at least one word that is padded. If this happens then γ_1 must always be the empty word.

In this case we have that:

$\{\beta_1^{rev}\#\alpha_2^{rev} : (\alpha_2^{rev},\beta_1)\delta \in C_{p_1,p_2,t}\}$ is of type $A_h^*\#_y A_p^*$;

$\{\alpha_1^{rev}\#\gamma_1 : (\alpha_1,\gamma_1\delta^L \in D_{p_1,p_3,t}\}$ is of type $A_h^*\#_y\epsilon_p^*$;

$\{\beta_2^{rev}\#\gamma_2 : (\beta_2,\gamma_2)\delta^R \in E_{p_2,p_3,t}\}$ is of type $A_h^*\#_y\epsilon_p^*A_p^*$.

Using Lemmas 14 and 15 again, we see that $K_{p_1,p_2,p_3,t}$ is of type $A_h^*A_h^*\#_y A_p^*A_h^*\#_y\epsilon_p^*A_p^*$.

In both of these two cases we have that $K_{p_1,p_2,p_3,t}$ is of type T_3. Using Lemma 14 we see that the set

$$\bigcup_{p_1,p_2,p_3\in Q, t\in\Gamma'} K_{p_1,p_2,p_3,t}$$

is of type T_3, so that the set

$$\{\beta^{rev}\#\alpha^{rev}\#\gamma : \alpha,\beta,\gamma \in L, \alpha\beta = \gamma\}$$

is of type T_3.

We have shown that, if (A, L) is a T_2-hyperbolic structure for M, then (A, L^{rev}) is a T_3-hyperbolic structure for M^{rev}. A similar argument establishes the converse, and so we have established Lemma 23. □

Given Lemma 25, we now have the following:

Lemma 27. *Let (A, L) be a T_2-hyperbolic structure of a monoid M. If $\zeta \in A^*$ with $|\zeta| = n$, then $\lambda \in L$ with $\lambda = \zeta$ can be calculated in time $\mathcal{O}(n \log n)$.*

Proof. We split ζ into two words ζ_1 and ζ_2 of length at most $\lceil|\zeta|/2\rceil$ and construct $\lambda_1, \lambda_2 \in L$ with $\lambda_1 = \zeta_1$ and $\lambda_2 = \zeta_2$ recursively. By Lemma 25 we can construct λ from λ_1 and λ_2 in time $O(|\lambda_1| + |\lambda_2|)$, and hence find λ in time $\mathcal{O}(n \log n)$. □

The last step in solving the word problem is given by the following result:

Lemma 28. *Let (A, L) be a T_2-hyperbolic structure for a monoid M and $\beta \in L$; then the set*

$$\{(\alpha,\gamma)\delta^L : \alpha,\gamma \in L, \alpha\beta = \gamma\}$$

is regular.

Proof. We will continue the notation used above. We are interested in the set S of all

$$(\mu, \nu, p, q, t) \in A^* \times A^* \times Q \times Q \times \Gamma'$$

such that

$$(p, t\sigma) \xrightarrow[A_{\mathrm{h}}^k \#_{\mathrm{y}} A_{\mathrm{p}}^k A_{\mathrm{h}}^l \#_{\mathrm{y}} A_{\mathrm{p}}^{l-m} \epsilon_{\mathrm{p}}^m]{\mu \# \beta \# \nu^{rev}} (q, t\sigma)$$

is a transition in P for some $k, l, m \in \mathbb{N}$ and some $\sigma \in \Gamma^*$.

Since β is fixed and $k, l, m \leqslant |\beta|$, the set S is finite. As described above the finite state automaton M_U reads words over $A(2, \$)$ and $\tau(s_U, (\alpha_1, \gamma_1)\delta^L) = U_{\alpha_1, \gamma_1}$. We construct a new finite state automaton M' by adding a state f (the only accept state of M') and transitions

$$\left\{ x \xrightarrow{(\mu, \nu)\delta^R} f : \text{there exists } (p, q, t) \in x \text{ and } (\mu, \nu, p, q, t) \in S \right\}.$$

Note that words accepted by M' are generally padded at the left, but a bounded number of padding symbols can appear on the right (due to the transitions to f). Let $(a_1, c_1) \dots (a_n, c_n) \in L(M')$; let α be the word resulting from $a_1 \dots a_n$ after removing all the padding symbols and let γ be the analogous word for $c_1 \dots c_n$. By the construction of M' there must exist $\alpha_1, \alpha_2, \gamma_1, \gamma_2 \in A^*$, $p, q \in Q$ and $t \in \Gamma'$ with $\alpha_1\alpha_2 \equiv \alpha$, $\gamma_1\gamma_2 \equiv \gamma$, $(p, q, t) \in U_{(\alpha_1, \gamma_1)\delta^L}$ and

$$(q_0, \perp) \xrightarrow{\alpha_1} (p, t\sigma) \xrightarrow{\alpha_2 \# \beta \# \gamma_2^{rev}} (q, t\sigma) \xrightarrow{\gamma_1^{rev}} (q_f, \perp)$$

for some $q_f \in F, \sigma \in \Gamma^*$. Hence $\alpha \# \beta \# \gamma^{rev} \in L(P)$ and $\alpha\beta = \gamma$ with $\alpha, \gamma \in L$.

Conversely, if $\alpha, \gamma \in L$ with $\alpha\beta = \gamma$, then $\alpha \# \beta \# \gamma^{rev}$ is in $L(P)$. So there exist $\alpha_1, \alpha_2, \gamma_1, \gamma_2 \in A^*$ with $\alpha_1\alpha_2 \equiv \alpha$, $\gamma_1\gamma_2 \equiv \gamma$, $|\alpha_2| = k$, $|\gamma_2| = l$, and either:

$$(q_0, \perp) \xrightarrow[A_{\mathrm{h}}^*]{\alpha_1} (p, t\sigma) \xrightarrow[A_{\mathrm{h}}^k \#_{\mathrm{y}} A_{\mathrm{p}}^k A_{\mathrm{h}}^l \# A_{\mathrm{p}}^l]{\alpha_2 \# \beta \# \gamma_2^{rev}} (q, t\sigma) \xrightarrow[A_{\mathrm{p}}^*(\epsilon_{\mathrm{p}}^* \cup A_{\mathrm{y}}^*)]{\gamma_1^{rev}} (q_f, \perp)$$

$$\text{or: } (q_0, \perp) \xrightarrow[A_{\mathrm{h}}^*]{\alpha_1} (p, t\sigma) \xrightarrow[A_{\mathrm{h}}^k \#_{\mathrm{y}} A_{\mathrm{p}}^k A_{\mathrm{h}}^l \# A_{\mathrm{p}}^{l-m} \epsilon_{\mathrm{p}}^m]{\alpha_2 \# \beta \# \gamma_2^{rev}} (q, t\sigma) \xrightarrow[\epsilon_{\mathrm{p}}^*]{\gamma_1^{rev}} (q_f, \perp)$$

for some $m > 0$, $p, q \in Q$, $t \in \Gamma'$, $q_f \in F$ and $\sigma \in \Gamma^*$; note that γ_1 must be the empty word in the second case. By the construction of M' the word $(\alpha_1, \gamma_1)\delta^L(\alpha_2, \gamma_2)\delta^R$ is accepted by M'. Using Lemma 4 we see that the set

$$\{(\alpha, \gamma)\delta^L : \alpha, \gamma \in L, \alpha\beta = \gamma\}$$

is regular. □

In a similar vein, one can prove:

Lemma 29. *Let (A, L) be a T_2-hyperbolic structure for a monoid M and $\alpha \in L$; then the set*

$$\{(\beta, \gamma)\delta^L : \beta, \gamma \in L, \alpha\beta = \gamma\}$$

is regular.

Given Lemma 28, we can check whether two given words in L represent the same element of M in linear time by choosing β to represent the identity element. Given Lemma 27, we now have the following result:

Theorem 30. *The word problem of a T_2-hyperbolic monoid is solvable in time $\mathcal{O}(n \log n)$.*

Given Lemma 23 we also have:

Theorem 31. *The word problem of a T_3-hyperbolic monoid is solvable in time $\mathcal{O}(n \log n)$.*

By Lemma 21, we then have:

Corollary 32. *The word problem of a T_1-hyperbolic monoid is solvable in time $\mathcal{O}(n \log n)$.*

6. Connections with biautomaticity

Lemmas 28 and 29 give that, for any given element m in a monoid M with a T_2-hyperbolic structure (A, L), the sets

$$ {}^{\$}_{m}L = \{(\alpha, \beta)\delta^L : m\alpha = \beta\} \quad \text{and} \quad {}^{\$}L_m = \{(\alpha, \beta)\delta^L : \alpha m = \beta\} $$

are regular. Since, by the definition of a T_2-hyperbolic structure, L is regular and L maps onto M, we have the following.

Theorem 33. *If M is a monoid with a T_2-hyperbolic structure (A, L) then (A, L) is also a left-biautomatic structure for M.*

We note that, by Lemma 23, we have:

Theorem 34. *If M is a monoid with a T_3-hyperbolic structure (A, L) then (A, L) is also a right-biautomatic structure for M.*

Given Lemmas 21 and 22, we then have that:

Corollary 35. *If M is a monoid with a T_1-hyperbolic structure (A, L) then (A, L) is also both a right-biautomatic and a left-biautomatic structure for M.*

We finish with the following observation:

Proposition 36. *If (A, L) is a T_2-hyperbolic structure for a monoid M then there exists $K \subseteq L$ such that (A, K) is a T_2-hyperbolic structure for M and K maps bijectively to M.*

Proof. (A, L) is also a left-biautomatic structure for M by Theorem 33. By Remark 6 there exists a regular language $K \subseteq L$ such that K maps bijectively to M. The set $K\{\#\}K\{\#\}K^{rev}$ is clearly regular; by Lemma 16 the set

$$ K\{\#\}K\{\#\}K^{rev} \cap L_{hyp} = \{(\alpha\#\beta\#\gamma^{rev} : \alpha\beta = \gamma, \alpha, \beta, \gamma \in K\} = K_{hyp} $$

is also a context-free language of type T_2; hence (A, K) is also a T_2-hyperbolic structure for M and K maps bijectively to M as required. □

We have similar results to Proposition 36 for T_1-hyperbolic and T_3-hyperbolic structures:

Proposition 37. *If (A, L) is a T_3-hyperbolic structure for a monoid M then there exists $K \subseteq L$ such that (A, K) is a T_3-hyperbolic structure for M and K maps bijectively to M.*

Proposition 38. *If (A, L) is a T_1-hyperbolic structure for a monoid M then there exists $K \subseteq L$ such that (A, K) is a T_1-hyperbolic structure for M and K maps bijectively to M.*

Acknowledgements

This paper was written whilst the authors were on study leave from the University of Leicester and the support of the University in this regard is appreciated. In addition, the paper was completed whilst the authors were visiting Friedrich Otto in Kassel; they are very grateful to him for his hospitality and for his constructive comments on the paper. The careful reading of the paper and the constructive suggestions of the two referees were also appreciated. The authors would also like to thank Chen-Hui Chiu and Hilary Craig for all their help and encouragement.

References

[1] J.M. Alonso, T. Brady, D. Cooper, V. Ferlini, M. Lustig, M. Michalik, M. Shapiro, H. Short, Notes on word hyperbolic groups, in: E. Ghys, A. Haefliger, A. Verjovsky (Eds.), Group Theory from a Geometric Viewpoint, World Scientific, 1991, pp. 3–63.
[2] R. Alur, P. Madhusudan, Visibly pushdown languages, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, 2004, pp. 202–211.
[3] J. Berstel, Transductions and Context-free Languages, Teubner, 1979.
[4] C.M. Campbell, E.F. Robertson, N. Ruškuc, R.M. Thomas, Automatic semigroups, Theoret. Comput. Sci. 250 (2001) 365–391.
[5] A. Duncan, R.H. Gilman, Word hyperbolic semigroups, Math. Proc. Cambridge Philos. Soc. 136 (2004) 513–524.
[6] D.B.A. Epstein, J.W. Cannon, D.F. Holt, S. Levy, M.S. Paterson, W. Thurston, Word Processing in Groups, Jones and Barlett, 1992.
[7] D.B.A. Epstein, D.F. Holt, The linearity of the conjugacy problem in word-hyperbolic groups, Internat. J. Algebra Comput. 16 (2006) 287–305.
[8] R.H. Gilman, On the definition of word hyperbolic groups, Math. Z. 242 (2002) 529–541.
[9] M. Gromov, Hyperbolic groups, in: S.M. Gersten (Ed.), Essays in Group Theory, in: MSRI Publ., vol. 8, Springer-Verlag, 1987, pp. 75–263.
[10] M. Hoffmann, D. Kuske, F. Otto, R.M. Thomas, Some relatives of automatic and hyperbolic groups, in: G.M.S. Gomes, J.-E. Pin, P.V. Silva (Eds.), Semigroups, Algorithms, Automata and Languages, World Scientific, 2002, pp. 379–406.
[11] M. Hoffmann, R.M. Thomas, Notions of automaticity in semigroups, Semigroup Forum 66 (2003) 337–367.
[12] M. Hoffmann, R.M. Thomas, Biautomatic semigroups, in: M. Liśkiewicz, R. Reischuk (Eds.), 15th International Symposium on Fundamentals of Computation Theory 2005, Lübeck, Germany, in: LNCS, vol. 3623, Springer-Verlag, 2005, pp. 56–67.
[13] D.F. Holt, Word-hyperbolic groups have real-time word problem, Internat. J. Algebra Comput. 10 (2000) 221–227.
[14] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[15] J.F.P. Hudson, Regular rewrite systems and automatic structures, in: J. Almeida, G.M.S. Gomes, P.V. Silva (Eds.), Semigroups, Automata and Languages, World Scientific, 1998, pp. 145–152.
[16] F. Otto, A. Sattler-Klein, K. Madlener, Automatic monoids versus monoids with finite convergent presentations, in: T. Nipkow (Ed.), Rewriting Techniques and Applications — Proceedings RTA '98, in: LNCS, vol. 1379, Springer-Verlag, 1998, pp. 32–46.